Ministry of Communications and Information Technology

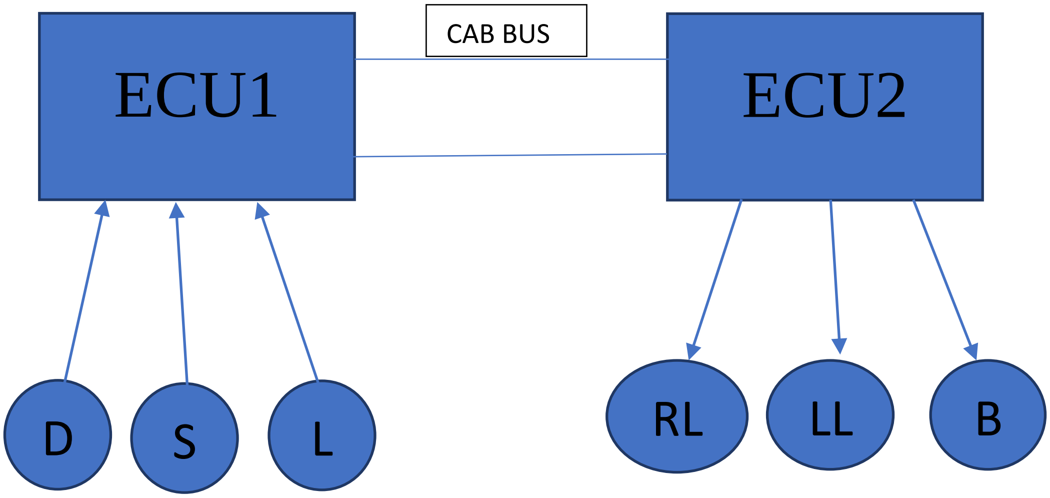
egypt fwd initiative

مستقبلنا رقمي

Automotive door control system design

Name: Omar Ahmed Dawood
Email: omarahmeddawood11@gmail.com

System schematic Diagram



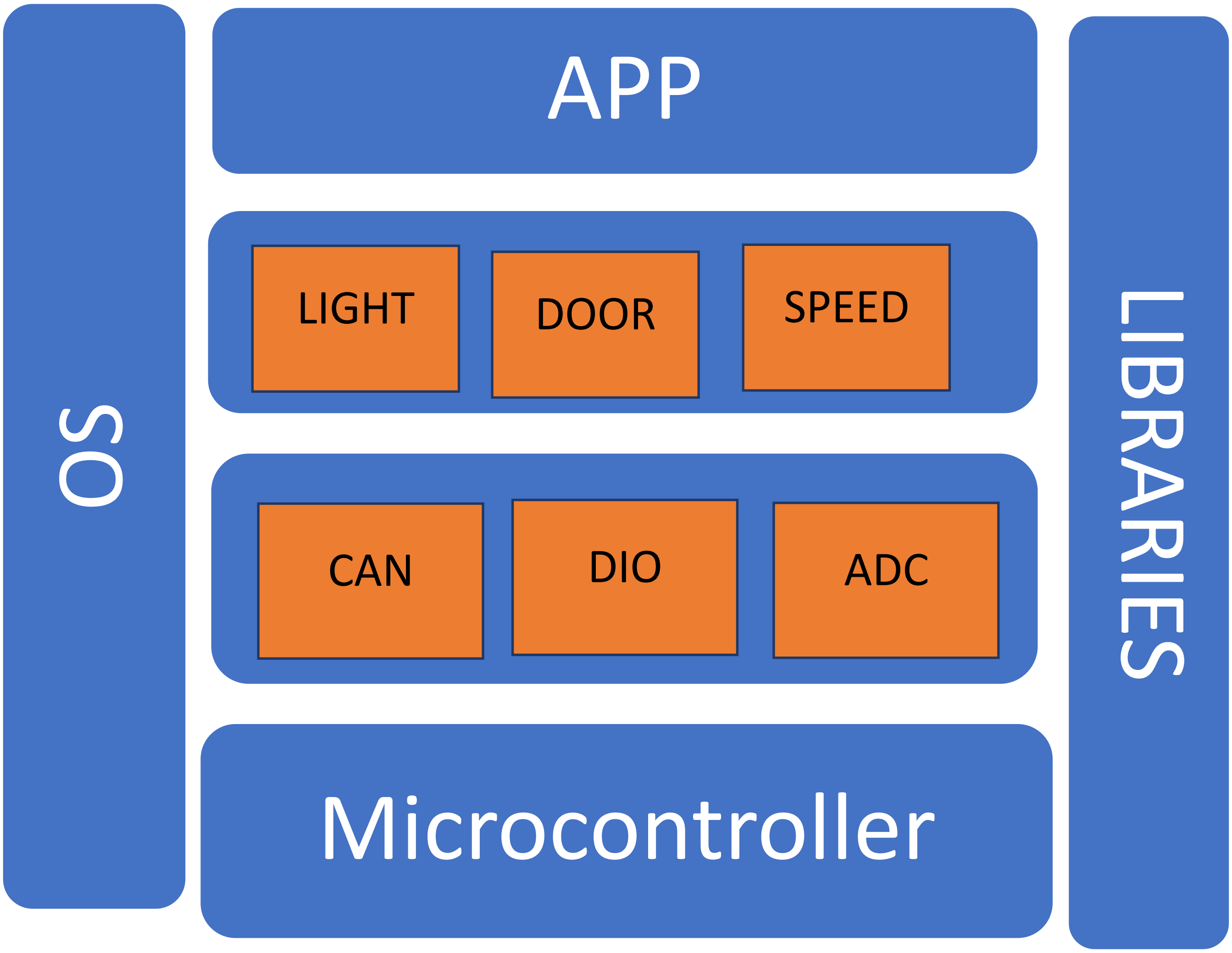
Software Requirements:

- Door status will be sent every 10ms.
- Light status will be sent every 20ms.
- Speed status will be sent every 5ms.
- If the door is opened while the car is moving → Buzzer ON, Lights OFF.
- If the door is opened while the car is stopped → Buzzer OFF, Lights ON.
- If the door is closed while the lights were ON → Lights are OFF after 3 seconds.
- If the car is moving and the light switch is pressed → Buzzer OFF, Lights ON.
- If the car is stopped and the light switch is pressed → Buzzer ON, Lights ON.

Static Design

For ECU1:

layered architecture:



Components and modules:

Modules:

- **MCAL**
  - ➢ CAN
  - ➢ DIO
  - ➢ ADC
- **HAL**
  - ➢ LIGHT
  - ➢ DOOR
  - ➢ SPEED
- **Communication Layer**
  - ➢ BCM
- **Sharable Layers**
  - ➢ OS
  - ➢ Libraries

Components:

- Door sensor
- Light sensor
- Speed sensor

APIS:

- CAN
  - Void CAN_INIT ()
  - Void CAN_Send (u32 data)
  - U32 CAN_recieve ()
- DIO
  - Void DIO_INIT ()
  - Void DIO_DirectionSet (u8 DIR)
  - Void DIO_PINSet (u8 STATE)
  - U8 DIO_GETPIN ()
- ADC
  - Void ADC_INIT ()
  - U16 ADC_StartConversion (u16 data)
- LIGHT
  - U8 LIGHT_STATE ()
- DOOR
  - U8 DOOR_STATE ()
- SPEED
  - U16 SPEED_RPM ()
- BCM
  - Void BCM_INIT ()
  - Void BCM_Send (u32 data)
  - U32 BCM_recieve ()

Description of functions:

```
➢ Void CAN_INIT ()
  {
          /* Init CAN protocol and bus frame and be able to send
          or receive.
           no return and no argument */
  }
➢ Void CAN_Send (u32 data)
  {
          /* start to send data on bus
          No return and take one argument that data I want to
          send it and its type unsigned int.
          */
  }
➢ U32 CAN_recieve ()
  {
          /* start to recieve data from bus
          return value of data that I receive it and its type
          unsigned int and no arguments.
          */
  }
➢ Void DIO_INIT ()
  {
          /* start DIO in the microcontroller to work and no
          return and no argument.
          */
  }
➢ Void DIO_DirectionSet (u8 DIR)
  {
          /* Set direction for pin and take direction as an
          argument and its type unsigned char and no return
          */ }
```

- Void DIO_PINSet (u8 STATE)

  ```
  {
          /* Set pin and take state of pin as argument and its type
          unsigned char and no return.
          */
  }
  ```

- U8 DIO_GETPIN ()

  ```
  {
          /* read pin and return its state as a return value its type
          unsigned char and no arguments.
          */
  }
  ```

- Void ADC_INIT ()

  ```
  {
          /* initialize ADC peripheral of the microcontroller
          To able to convert and no return and no arguments.
          */
  }
  ```

- U16 ADC_StartConversion (u16 data)

  ```
  {
          /* start conversion and take data that I want to convert
          it as an argument and its type unsigned short and return
          its digital value that its type unsigned short.
          */
  }
  ```

- U8 LIGHT_STATE ()

  ```
  {
          /* read Light Switch State and return it
          Its return type unsigned char.
          */
  }
  ```

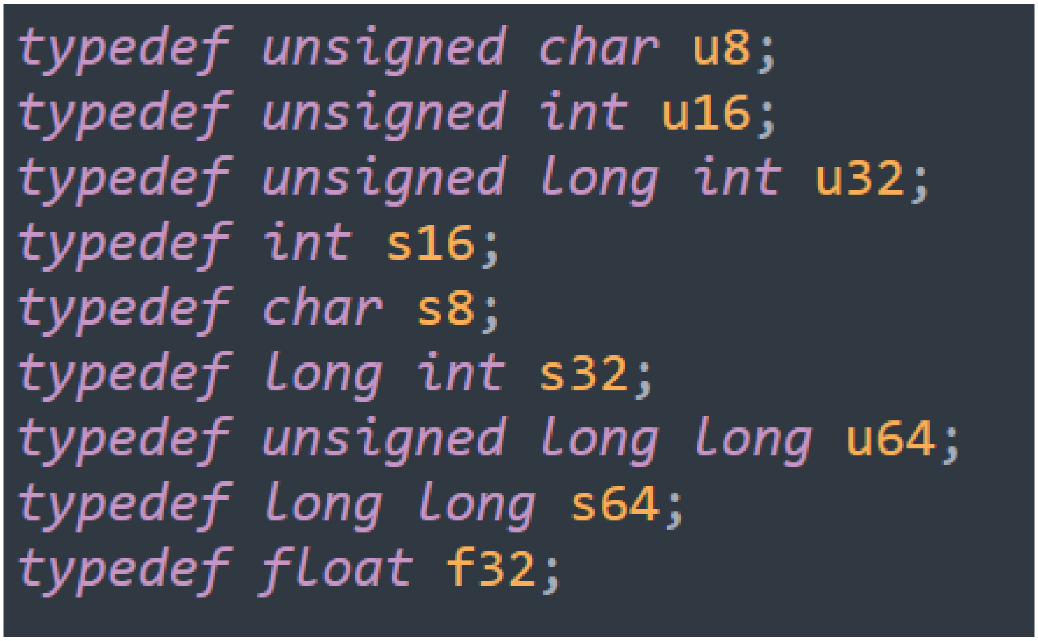
- U8 DOOR_STATE ()

```
{
        /* read door sensor State and return it
        Its return type unsigned char.
*/
}
```

- U16 SPEED_RPM ()

```
{
        /* read speed from speed sensor and return the
        corresponding volt.
        Its return type unsigned short.
*/
}
```

Used Typedef



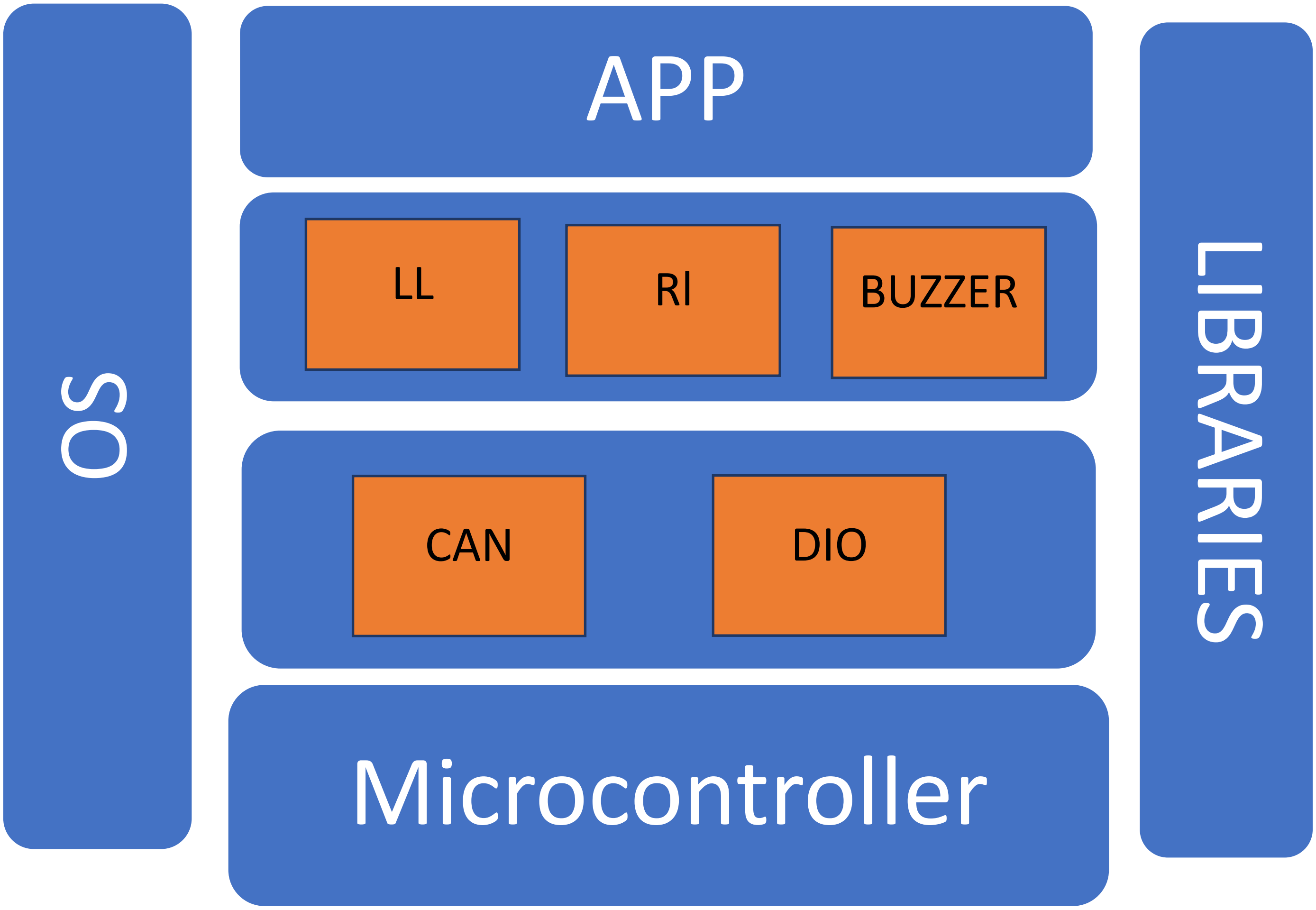
```c
typedef unsigned char u8;
typedef unsigned int u16;
typedef unsigned long int u32;
typedef int s16;
typedef char s8;
typedef long int s32;
typedef unsigned long long u64;
typedef long long s64;
typedef float f32;
```

Folder Structure

- Project
  - **MCAL**
    - CAN
    - DIO
    - ADC
      - **HAL**
        - DOOR
        - LIGHT
        - SPEED
          - **APP**
            - APP
              - **Main**
                - Main ()

For ECU2:

layered architecture:



Components and modules:

Modules:

- **MCAL**
  - ➢ CAN
  - ➢ DIO
- **HAL**
  - ➢ Left Light
  - ➢ Left Light
  - ➢ Buzzer
- **Communication Layer**
  - ➢ BCM
- **Sharable Layers**
  - ➢ OS
  - ➢ Libraries

Components:

- Left Light
- Right Light
- Buzzer

APIS:

- CAN
  - Void CAN_INIT ()
  - Void CAN_Send (u32 data)
  - U32 CAN_recieve ()
- DIO
  - Void DIO_INIT ()
  - Void DIO_DirectionSet (u8 DIR)
  - Void DIO_PINSet (u8 STATE)
  - U8 DIO_GETPIN ()
- Left LIGHT
  - Void Left_ LIGHT_ON ()
  - Void Left_ LIGHT _OFF ()
- Buzzer
  - Void Buzzer_ON ()
  - Void Buzzer_OFF ()
- Right Light
  - Void Right_ LIGHT_OFF ()
  - Void Right_ LIGHT_ON ()
- BCM
  - Void BCM_INIT ()
  - Void BCM_Send (u32 data)
  - U32 BCM_recieve ()

Description of functions:

```
➢ Void CAN_INIT ()
  {
        /* Init CAN protocol and bus frame and be able to send
        or receive.
         no return and no argument */
  }
➢ Void CAN_Send (u32 data)
  {
        /* start to send data on bus
        No return and take one argument that data I want to
        send it and its type unsigned int.
        */
  }
➢ U32 CAN_recieve ()
  {
        /* start to recieve data from bus
        return value of data that I receive it and its type
        unsigned int and no arguments.
        */
  }
➢ Void DIO_INIT ()
  {
        /* start DIO in the microcontroller to work and no
        return and no argument.
        */
  }
➢ Void DIO_DirectionSet (u8 DIR)
  {
        /* Set direction for pin and take direction as an
        argument and its type unsigned char and no return
        */ }
```

```
➢ Void DIO_PINSet (u8 STATE)
  {
        /* Set pin and take state of pin as argument and its type
        unsigned char and no return.
        */
  }
➢ U8 DIO_GETPIN ()
  {
        /* read pin and return its state as a return value its type
        unsigned char and no arguments.
        */
  }
➢ Void Left_ LIGHT_ON ()
  {
        /* set high to the pin of Left Light and no argument and
        no return.
        */
  }

➢ Void Left_ LIGHT_OFF ()
  {
        /* Clear the pin of Left Light and no argument and no
        return.
        */
  }

➢ Void Right_ LIGHT_ON ()
  {
        /* set high to the pin of Right Light and no argument
        and no return.
        */
  */ }
```

- Void Right_ LIGHT_OFF ()
  {

      /* Clear the pin of Right Light and no argument and no return.
      */

  }
- Void Buzzer_ON ()
  {

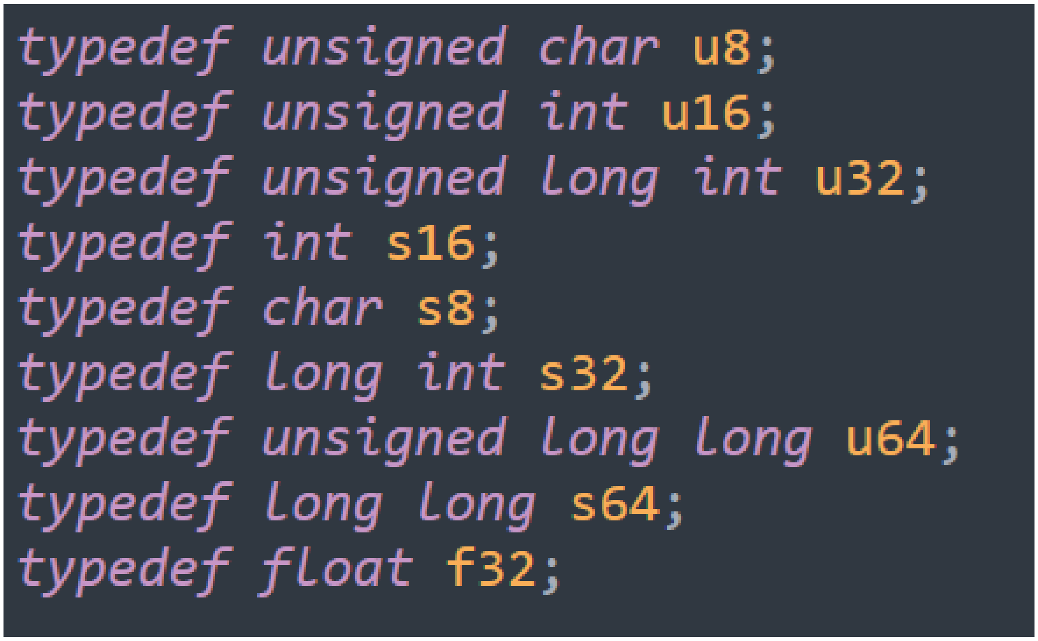
      /* set high to the pin of Buzzer and no argument and no return.
      */

  }
- Void Buzzer_OFF ()
  {

      /* Clear the pin of Buzzer and no argument and no return.
      */

  }

- **Used Typedef**



```
typedef unsigned char u8;
typedef unsigned int u16;
typedef unsigned long int u32;
typedef int s16;
typedef char s8;
typedef long int s32;
typedef unsigned long long u64;
typedef long long s64;
typedef float f32;
```

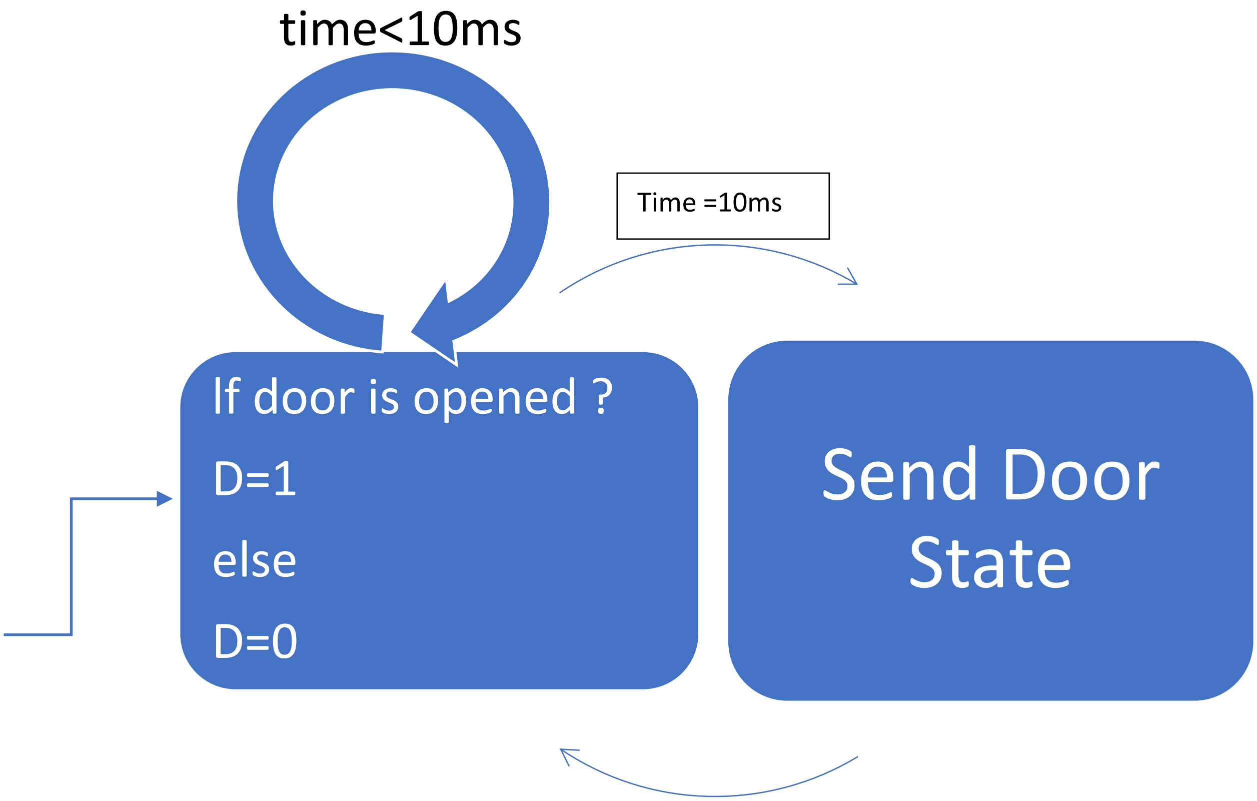
Folder Structure

- Project
  - **MCAL**
    - ➢ CAN
    - ➢ DIO
      - **HAL**
        - o Buzzer
        - o Left LIGHT
        - o Right Light
          - **APP**
            - ❖ APP
              - **Main**
                - ✓ Main ()

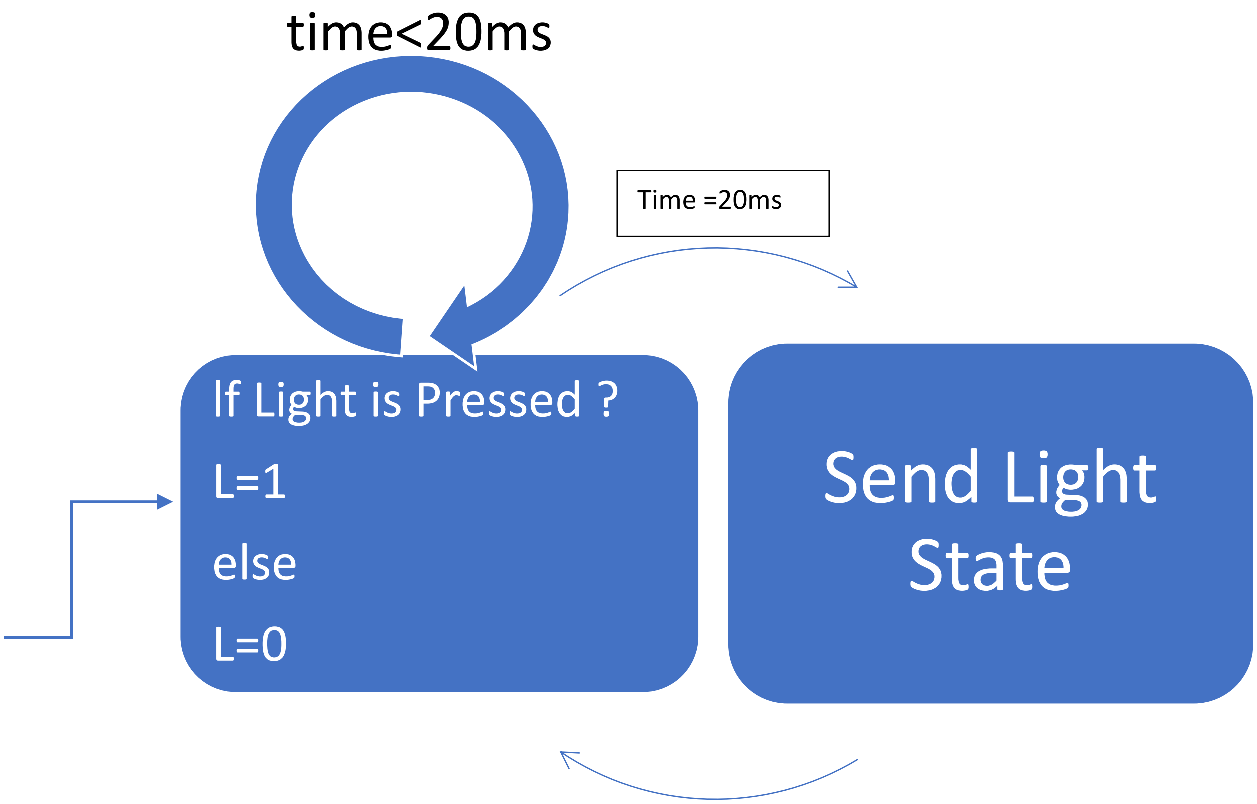
Dynamic Design

For ECU1

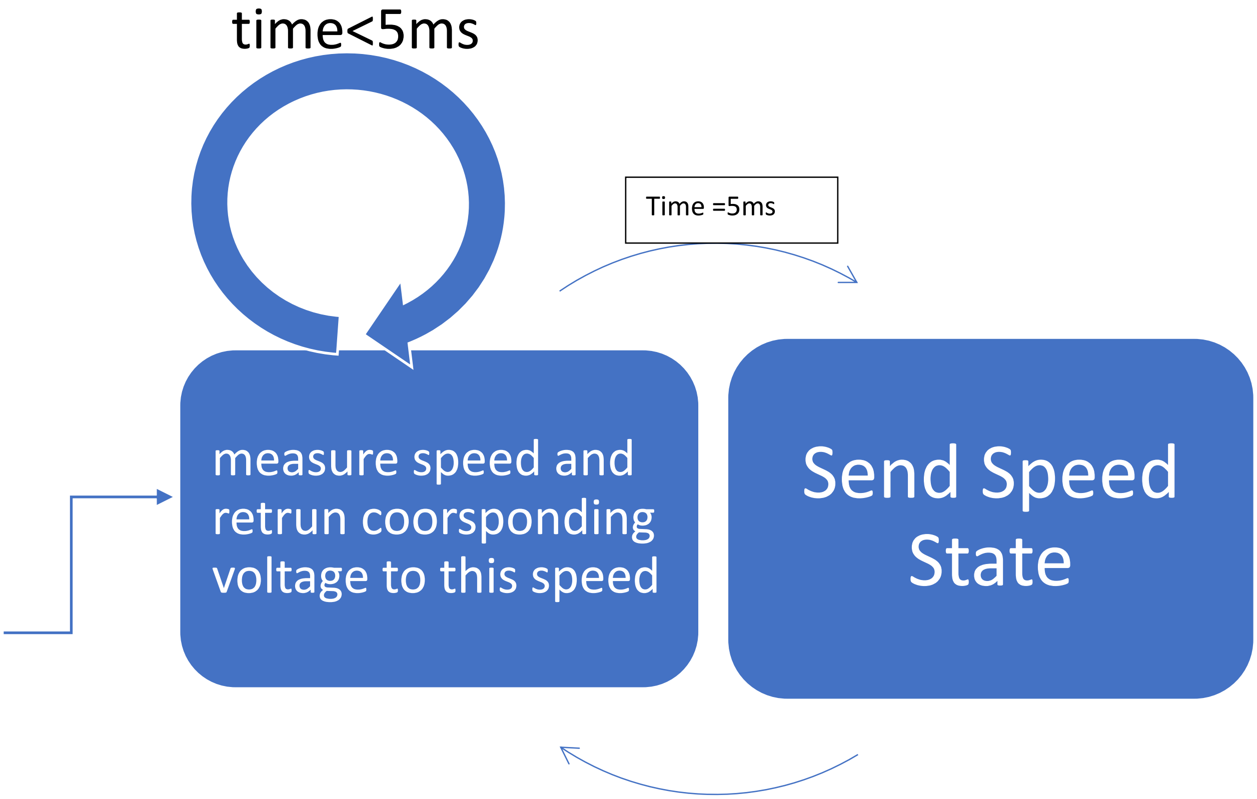
State Machine of door



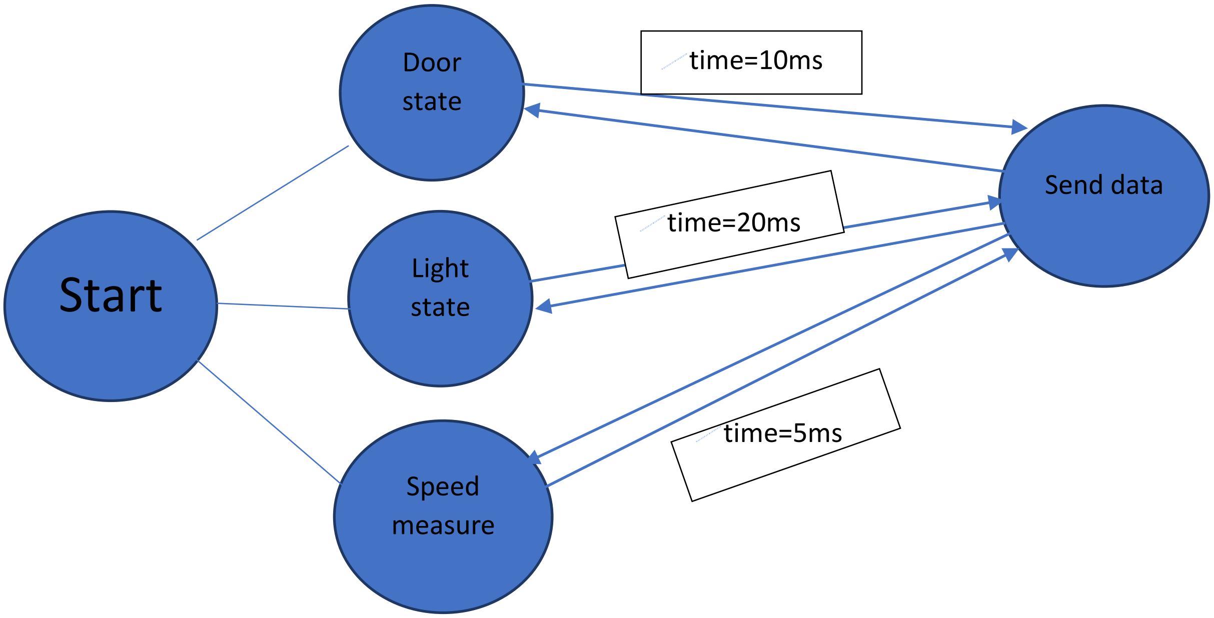
State Machine of Light



State Machine of speed



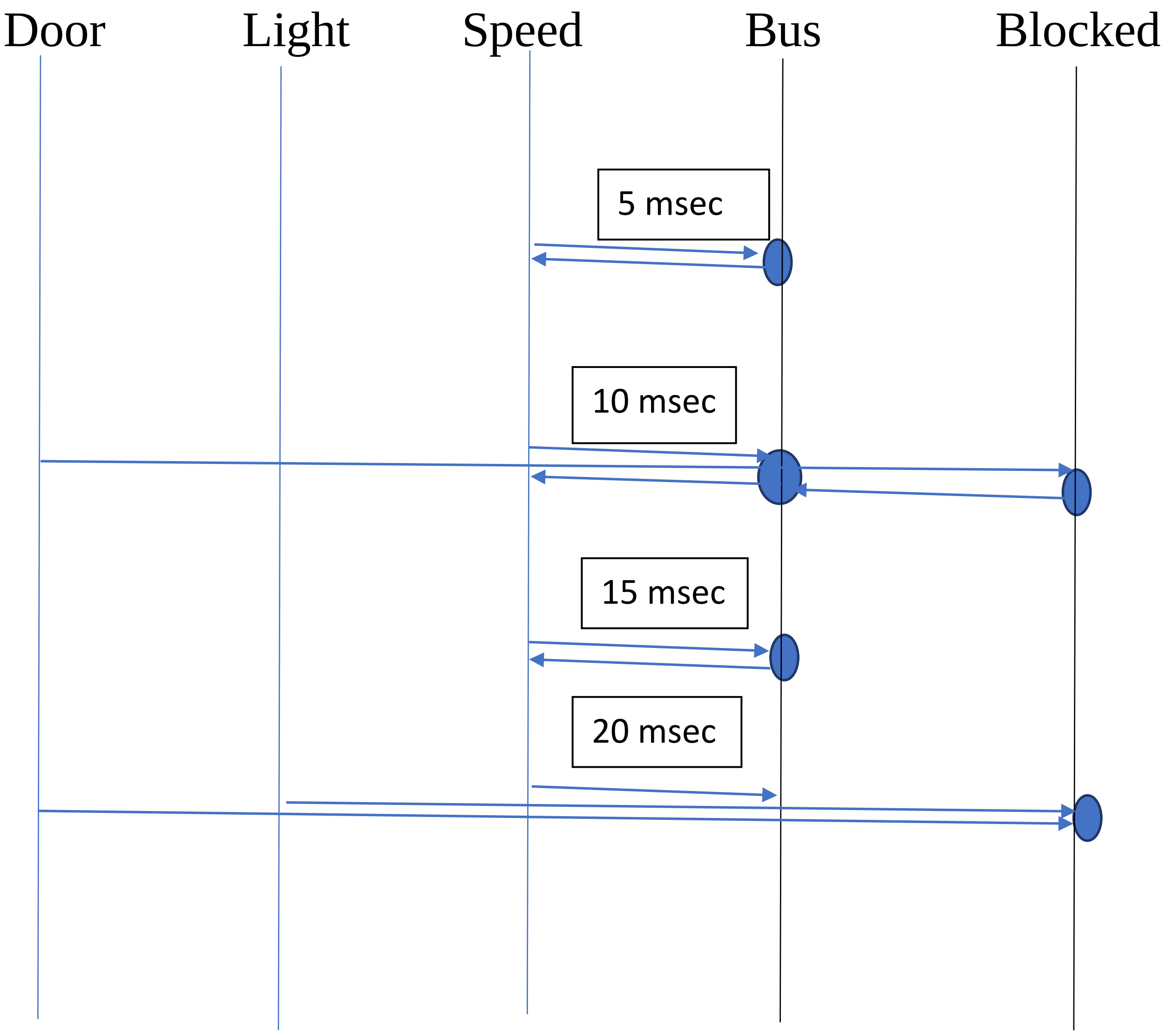
State Machine of operation:



Note

- If two states must send in same time together, in this case the highest priority will send first.

Sequence diagram



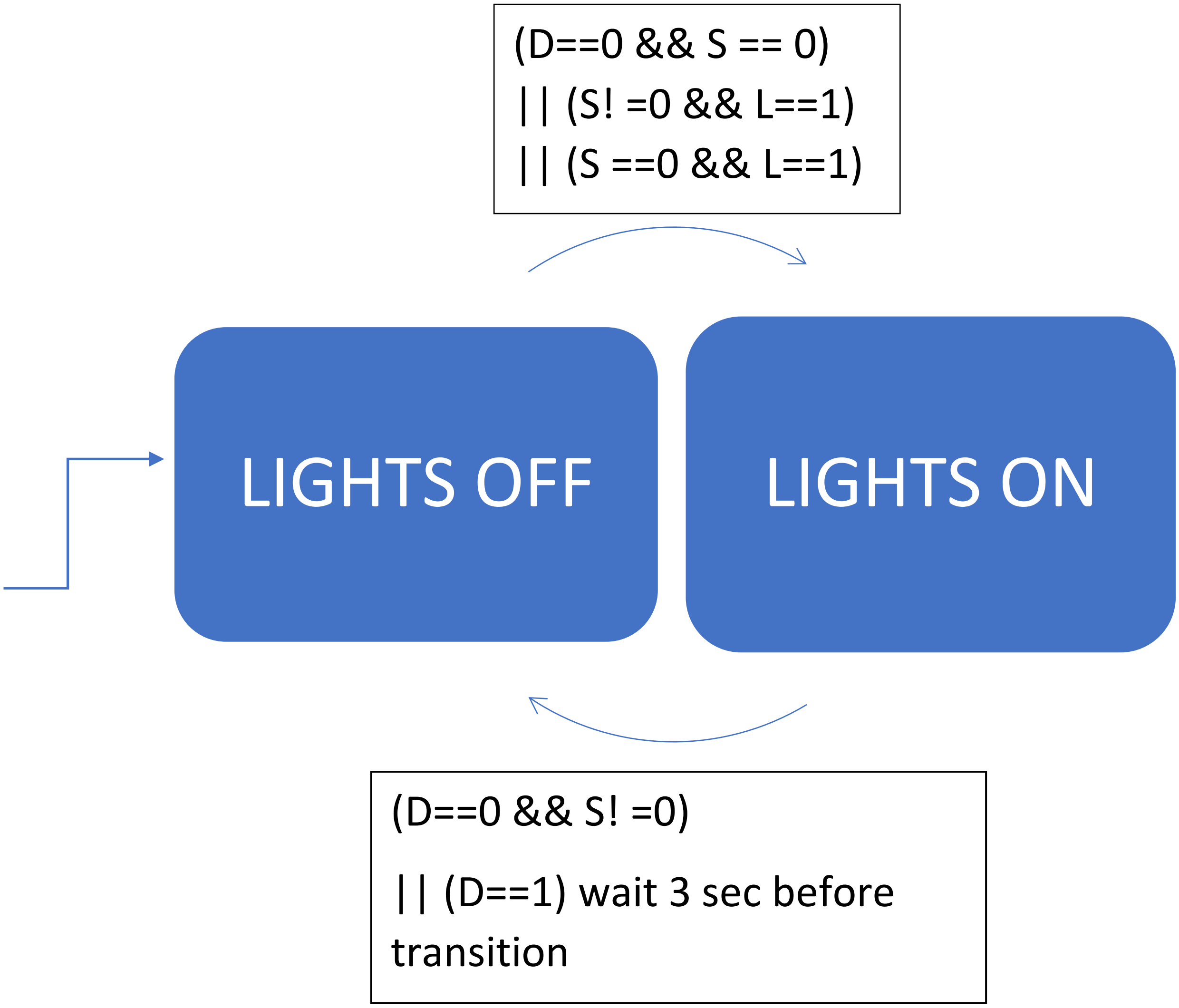
CPU LOAD:

Assume Execution Time for all tasks = 2 msec

CPU Load = (6/20) = 0.3 = 30 %

For ECU2

State Machine of LL and RL

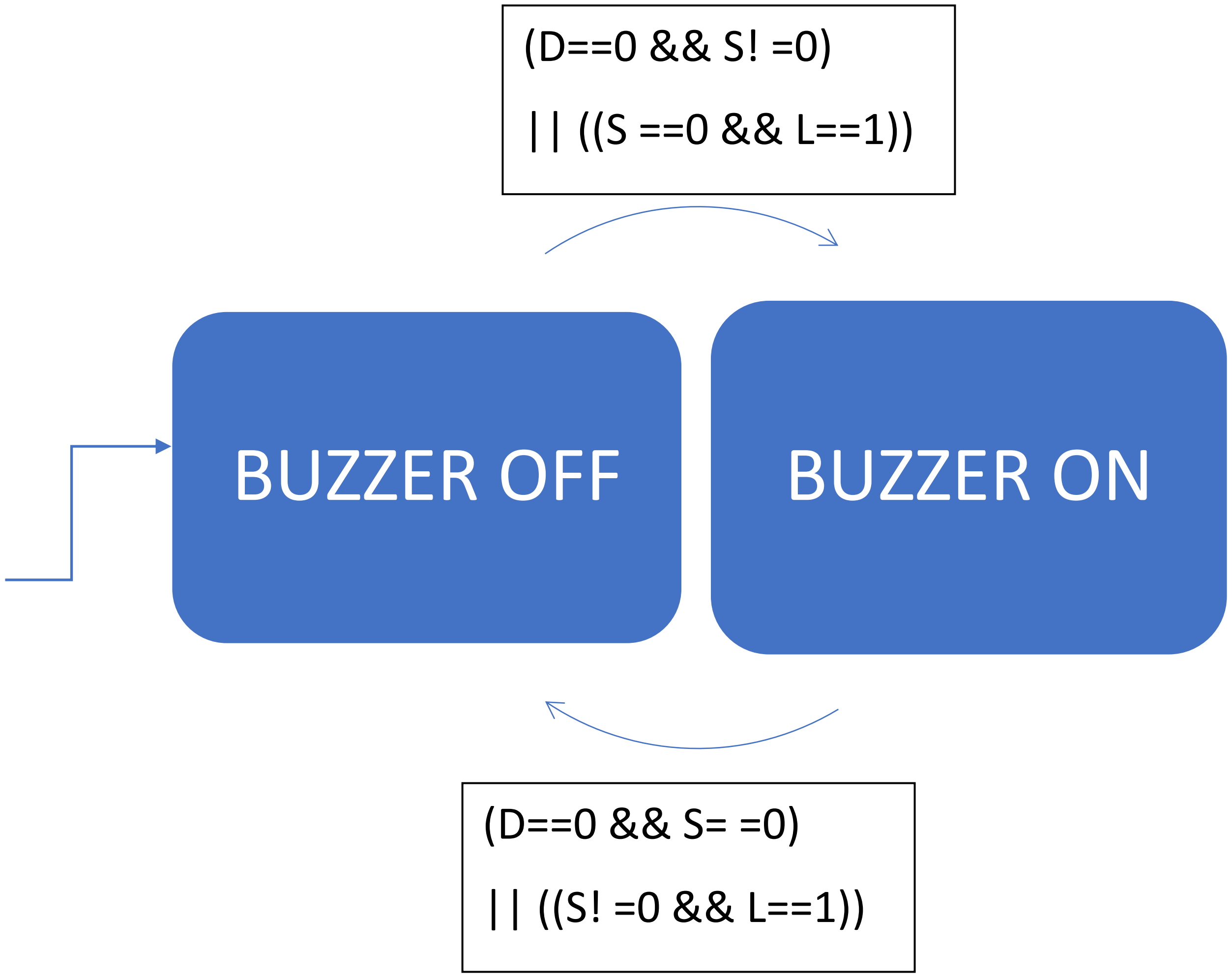


(D==0 && S == 0)
|| (S! =0 && L==1)
|| (S ==0 && L==1)

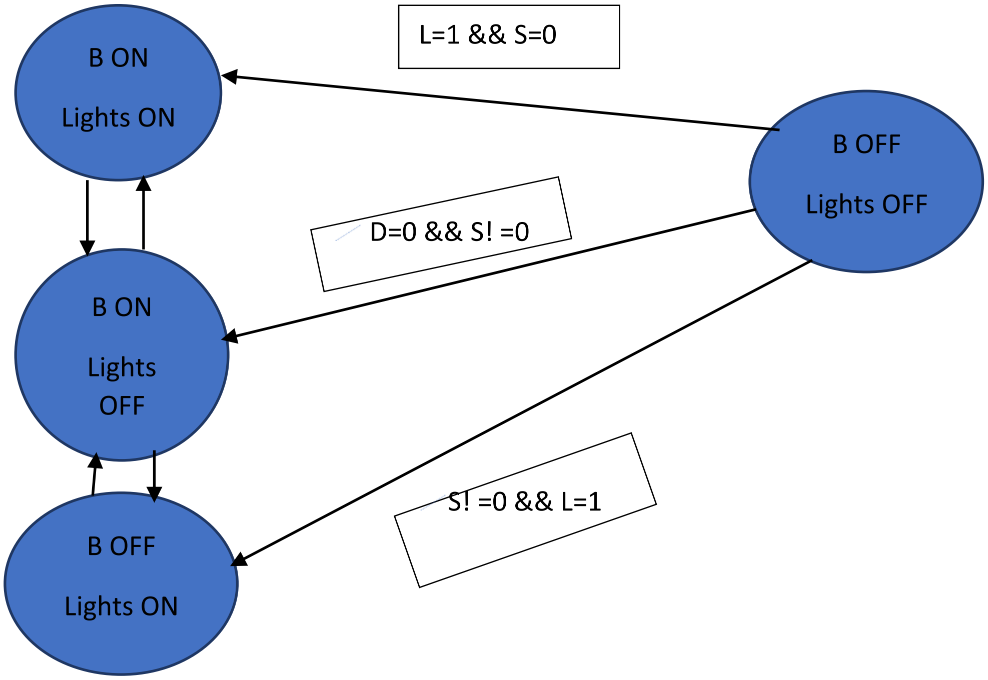
(D==0 && S! =0)

|| (D==1) wait 3 sec before transition

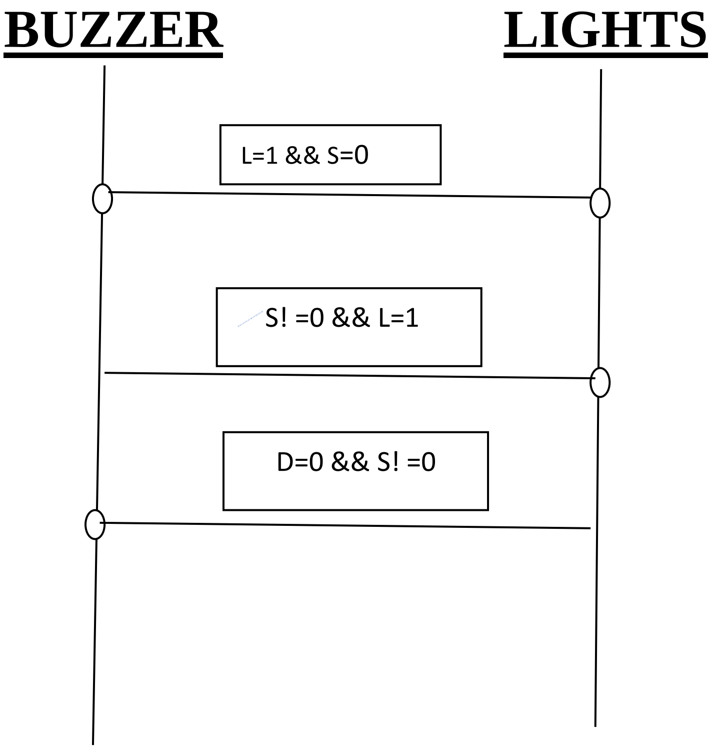
State Machine of BUZZER



State Machine of operation:



Sequence diagram



CPU LOAD:

Assume Execution Time for all tasks = 2 sec

CPU Load = (6/20) = 0.3 = 30 %

BUS LOAD:

BUS LOAD = (1/5Sec) = 20 %